UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LENGUAJES FORMALES Y DE PROGRAMACIÓN
SEGUNDO SEMESTRE 2,021.

Sección	Catedrático	Tutor académico
A +	Ing. Otto Amilcar Rodríguez Acosta	Fernando Feliciano Chajón del Cid
B +	Ing. David Estuardo Morales Ajcot	Bryan Gustavo López Echeverría
A -	Inga. Damaris Campos de López	César David Juárez González
B -	Inga. Zulma Karina Aguirre Ordoñez	Douglas Omar Arreola Martínez

---

PROYECTO 1

Objetivos

Que el estudiante:

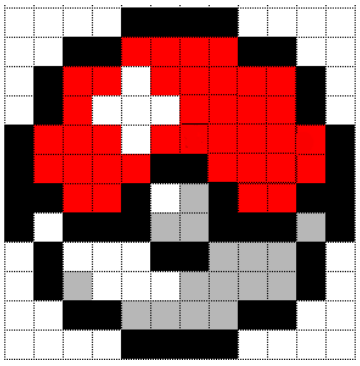
- ➔ Desarrolle una solución de software implementando un analizador léxico mediante autómatas.
- ➔ Aplique los conocimientos adquiridos en el laboratorio, del lenguaje de programación Python
- ➔ Desarrolle una interfaz gráfica utilizando el lenguaje Python.

Descripción

La empresa *Bitxelart* se dedica a la elaboración de imágenes digitales en estilo **pixel art**, para realizar estas piezas digitales un empleado toma una imagen que utilizará como referencia y la divide en una cuadrícula, anotando en un **archivo de texto**, con extensión **pxla**, las características de la misma (color, dimensiones, etc), luego este archivo es trasladado a un diseñador digital que utilizando el lenguaje de marcado **HTML**, dibuja la imagen solicitada.

En los últimos meses la demanda de estas piezas se ha multiplicado y al diseñador se le hace imposible terminar todos los trabajos encargados a tiempo, por lo que se le contrata a usted para que realice un programa que sea capaz de leer los archivos de texto de estas piezas y genere las imágenes correspondientes, para así automatizar este proceso y evitar pérdidas a la empresa.

Un ejemplo de estas imágenes es el siguiente:



Archivo de texto:

El archivo se compone de 7 secciones, la estructura es la siguiente:

```
TITULO="Pokebola";
ANCHO=300;
ALTO=300;
FILAS=12;
COLUMNAS=12;
CELDAS = {
       [0,0,FALSE,#000000],
       [0,1,FALSE,#000000],
       [3,3,FALSE,#000000],
       [3,4,TRUE,#000000],
       [3,5,TRUE,#000000],
       [3,6,TRUE,#000000],
       [3,7,TRUE,#000000],
       [4,1,FALSE,#000000]
   };
FILTROS = MIRRORX;

@@@@

TITULO="Estrella";
ANCHO=300;
ALTO=300;
FILAS=4;
COLUMNAS=4;
CELDAS = {
       [0,0,FALSE,#000000],
       [1,1,FALSE,#000000],
       [3,3,FALSE,#000000],
       [2,1,FALSE,#000000]
   };
FILTROS = MIRRORX,MIRRORY,DOUBLEMIRROR;
```

1. TITULO:
   Esta sección indica el título de la imagen que se generará, se conforma de la palabra reservada *TITULO* seguida de un signo igual (=) y una cadena de texto encerrada en comillas dobles (“) y finaliza con punto y coma (;).

2. ANCHO:
   Esta sección indica el ancho en píxeles que tendrá la imagen, la sección se conforma de la palabra reservada *ANCHO* seguida de un signo igual (=), como valor tomará un número entero positivo y finaliza con un punto y coma (;).

3. ALTO:
   Esta sección indica el alto en píxeles que tendrá la imagen, la sección se conforma de la palabra reservada *ALTO* seguida de un signo igual (=), como valor tomará un número entero positivo y finaliza con un punto y coma (;).
4. FILAS:
   Esta sección indica el número de filas en las que se divide la imagen, la sección se conforma de la palabra reservada *FILAS* seguida de un signo igual

(=), como valor tomará un número entero positivo y finaliza con un punto y coma (;).

5. COLUMNAS:
   Esta sección indica el número de columnas en las que se divide la imagen, la sección se conforma de la palabra reservada *COLUMNAS* seguida de un signo igual (=), como valor tomará un número entero positivo y finaliza con un punto y coma (;).

6. CELDAS:
   En esta sección se definen los colores que tomará cada uno de los "pixeles" en los que está dividido la imagen, la sección inicia con la palabra reservada *CELDAS* seguida de un signo igual (=), como valor tomará una lista de elementos que estará encerrada en un par de llaves y cada elemento separado por una coma (,).
   Los elementos de esta lista se encuentran encerrados en un par de corchetes y dentro de estos están dos valores separados por coma (,). El primer elemento es un valor entero positivo que representa la posición de la celda en el eje X, el segundo elemento es un valor entero positivo que representa la posición de la celda en el eje Y, el tercer valor es una cadena que representa un booleano (TRUE o FALSE) y por último un color representado en forma hexadecimal. Ejemplos:

   - [0,0,TRUE,#FA0000]
     True nos indica que la celda **sí** será pintada del color indicado, el color es #FA0000 que equivale al color rojo.

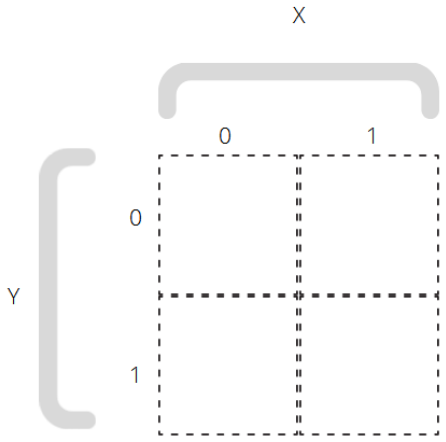
   - [0,1,FALSE,#FA0000]
     False nos indica que la celda **no** será pintada del color indicado, es decir el color es ignorado y la celda no se pintará.

** Si una posición no aparece no se pintará de ningún color, lo que es equivalente a que la celda tuviera el parámetro FALSE en ella.

La sección finaliza con un punto y coma(;).

Al finalizar la definición de una imagen puede aparecer un separador que consiste en una cadena de cuatro arrobas (**@@@@**), lo cual nos indica que en cada archivo es posible definir 'N' imágenes.

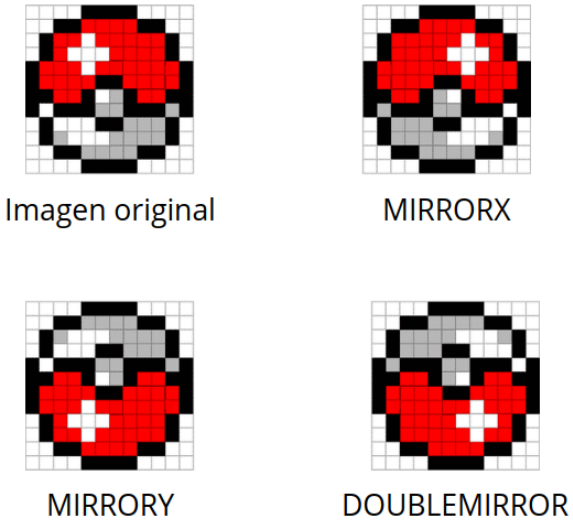
Nota: El tamaño de las celdas debe ser calculado por el estudiante.



7. FILTROS:

   Esta sección indica los filtros que podrán ser aplicados a esta imagen, la sección inicia con la palabra reservada *FILTROS* seguida de un signo igual (=), seguida de una lista de palabras reservadas separadas por una coma (,) la sección finaliza con un punto y coma (;). Las palabras reservadas pueden ser:

   - MIRRORX: Este filtro gira horizontalmente la imagen original.
   - MIRRORY: Este filtro gira verticalmente la imagen original.
   - DOUBLEMIRROR: Este filtro gira horizontal y verticalmente la imagen original.



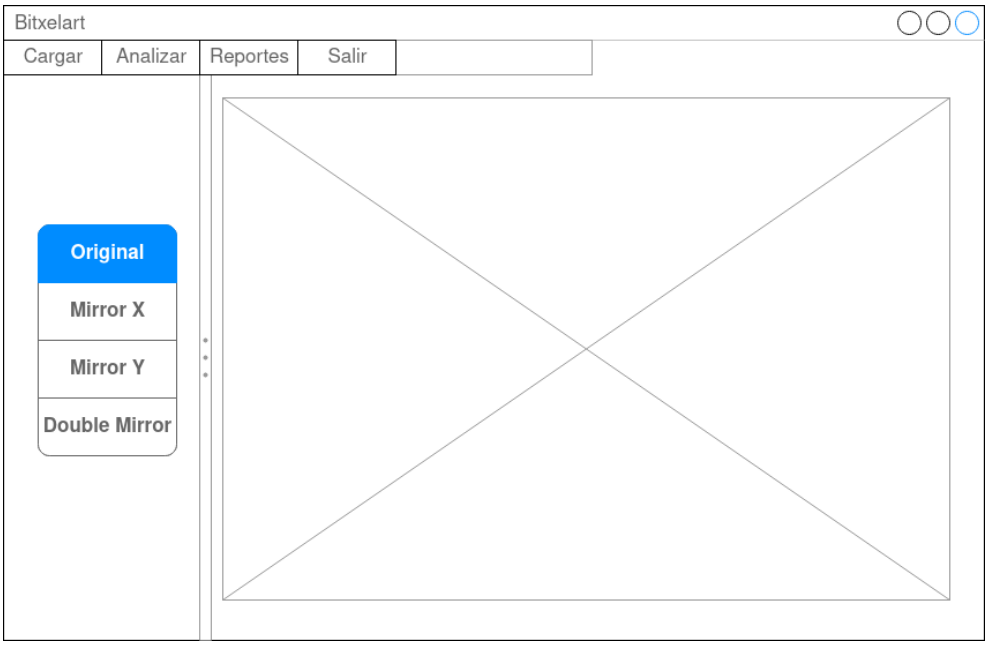
Funciones del sistema

La aplicación cuenta con un interfaz gráfica, en la cual se poseen las siguientes opciones:

1. **Cargar archivo:** Muestra una ventana emergente que permite al usuario seleccionar un archivo **pxla** y cargarlo a memoria.
2. **Analizar archivo:** Analiza el archivo cargado mediante un ***autómata definido e implementado por el estudiante*** y genera los archivos HTML tanto de la imagen original como de las imágenes con filtro. los datos obtenidos durante la ejecución del reporte.
3. **Ver reportes:** Se debe de escribir un archivo HTML con los datos del reporte generado, la manera en que los datos son mostrados deben de ser agradables al usuario. Los reportes deben abrirse automáticamente al seleccionar esta opción.
4. **Seleccionar imagen:** El programa debe contar con una lista de las imágenes analizadas en el archivo, se deberá poder seleccionar una imagen para poder visualizarla junto con los filtros aplicados en ella.
5. **Ver imagen:** El programa debe contar con un apartado para poder visualizar las imágenes generadas, para poder hacer esto se recomienda convertir el archivo HTML generado a un archivo con extensión jpg o png.
   a. Original: Muestra la imagen original sin aplicar ningún filtro.
   b. MirrorX: Muestra la imagen girada horizontalmente.
   c. MirrorY: Muestra la imagen girada verticalmente
   d. DoubleMirror: Muestra la imagen girada horizontal y verticalmente.

   Si se selecciona un filtro que no ha sido generado debe notificar al usuario.
6. **Salir:** Termina la ejecución de la aplicación.

Interfaz sugerida



Reportes

La aplicación debe generar reportes en formato HTML, los reportes son:

1. Reporte de tokens: Se debe generar una tabla con los tokens que se identificaron en el archivo de entrada, se debe indicar el token, lexema, fila y columna del token leído.
2. Reporte de errores: Se debe generar una tabla con los errores léxicos detectados durante el análisis, indicando la fila, columna y caracter que provocó el error.

Entregables

- ➔ Manual de usuario.
- ➔ Manual técnico, en este se debe incluir además del contenido habitual lo siguiente:
  - ◆ Tabla de tokens, en la cual se debe indicar el patrón (Expresión regular) que define a cada token.
  - ◆ El proceso del **método del árbol** realizado para generar su **autómata finito determinista** (Incluir también el grado de su autómata). Este autómata debe coincidir con el autómata implementado en su código fuente.
- ➔ Código fuente.

Consideraciones importantes

- ➔ El proyecto debe de desarrollarse individualmente.
- ➔ Se debe de utilizar el lenguaje de programación Python
- ➔ La entrega se realizará en la plataforma UEDI. Todos los archivos solicitados deberán ser entregados en un archivo comprimido zip con el siguiente nombre: **[LFP]Proyecto1_<<Carnet>>.zip**. Tomar en cuenta que el único medio de entrega es la plataforma UEDI.

- ➔ La calificación se realizará en línea y se grabará, esto para que quede constancia de la forma en que se calificó y como soporte en la toma de decisiones en reclamos por parte del alumno si se presenta el caso.
- ➔ La calificación es personal con una duración máxima de 30 minutos, en el horario posteriormente convenido. Durante la calificación el estudiante no podrá modificar el archivo de entrada ni el código fuente de su aplicación.
- ➔ El estudiante es responsable del horario que elija para calificarse, en caso de no poder presentarse deberá notificar al auxiliar con suficiente anticipación (2 días antes) para ceder su lugar a otro estudiante, en caso contrario el estudiante solo obtendrá el 80% de su nota obtenida.
- ➔ No se dará prórroga para la entrega del proyecto.
- ➔ **COPIA PARCIAL O TOTAL DE LA PRÁCTICA TENDRÁ UNA NOTA DE 0 PUNTOS, Y SE NOTIFICARÁ AL CATEDRÁTICO DEL CURSO Y POSTERIORMENTE SI SE REQUIERE A LA ESCUELA DE SISTEMAS PARA QUE SE APLIQUEN LAS SANCIONES CORRESPONDIENTES.**
- ➔ **Fecha de entrega del proyecto: 23 de septiembre de 2021 antes de las 23:59 horas.**